Randomness Properties of Cryptographic Hash Functions

Micah A. Thornton

Southern Methodist University *Bobby B. Lyle School of Engineering*

August 8, 2017

Outline

Outline

Primary Hypothesis

Hypothesis

Assuming a **cryptographic hash** is being used to increase the **apparent randomness** of a data set, It is possible to **formulate metrics** to choose the best hash for this purpose.

Conclusion

The hypothesis holds, and suitable metrics were formulated and verified.

Secondary Hypothesis

Secondary Hypothesis

The **A Posteriori** method described in this research is a valid approach for entropy extraction of a **weak random source** in the form of inter packet delays between packet arrivals.

Conclusion

The method proposed can indeed function as a randomness extractor on network timing data.

Cryptographic Hash Functions

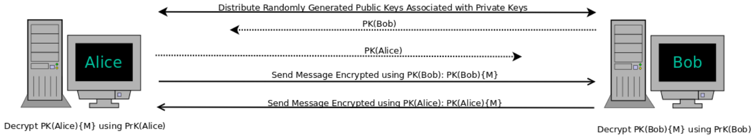
14 Common Cryptographic Hashes

Blake 2 32-bit(bl2s)	Blake 2 64-bit(bl2b)	MD5(md5)
SHA-1(s1)	SHA-2 224-bit(s2224)	SHA-2 512-bit(s2512)
SHA-2 256-bit(s256)	SHA-3 224-bit(s3224)	SHA-3 256-bit(s3256)
SHA-3 384-bit(s3384)	SHA-3 512-bit(s3512)	SHA-2 384-bit(s384)
shake 128-bit(ske128)	shake 256-bit(ske256)	

- Cryptographic hashes are used in many security applications.
- The bit size of the function represents the length of the output string.
- In this work, only portions of bit streams were fed to the hash function at a time, according to output length.

Modern Applications of Random Values

Example application of random values to public key cryptography



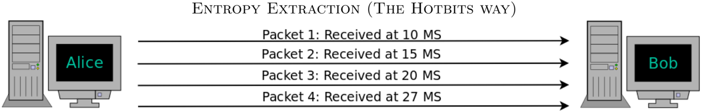
- In cryptography:
  - RSA: RNs are used to generate primes (No RNG specified)
  - 3-DES: RNs used as key-bundle (Specific RNG ANSI x9.31)
  - Blowfish: RN used a 52-bit key (No RNG specified)
  - Twofish: RN used as up to 256-bit key (No RNG specified)
  - AES: RNs used as key-IV-salt bundle (NIST specified RNG)
- In science:
  - Statistics: Taking random sample
  - Analysis: Extraction of signal from noise
  - Simulation: Providing a spectrum of inputs

Approaches to Random Generation

Giuseppe Lodovico Lagrangia

- Pseudo-Random Number Generators (PRNGs)
  - Shift Registers (LFSR, NLFSR) - Golomb (1948)
  - Linear Congruential Generators (LCG) - D. H. Lehmer (1949)
  - Blum Blum Shub (BBS) - Blum,Blum, and Shub (1986)
  - Mersenne Twister (MT) - Matsumoto & Nishimura (1997)
- True Random Number Generators (TRNGs)
  - Atmospheric Noise (random.org)
  - Radioactive Decay (hotbits.org)

Entropy Extractors



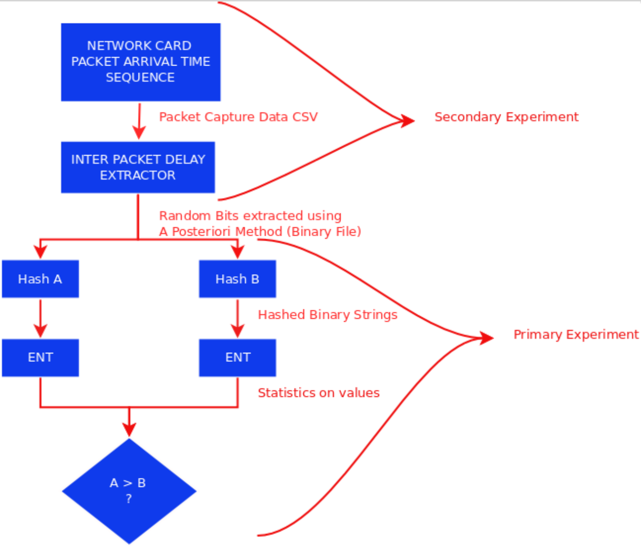
$$T_1 = P_2 - P_1 = 15 - 10 = 5$$

$$T_2 = P_4 - P_3 = 27 - 20 = 7$$

if T1 > T2:
    record one
if T1 < T2:
    record zero
if T1 = T2:
    record nothing

Outline

Process Flow



A Posteriori Extraction Method

Given X such that $X = \{x_1, x_2, x_3, ..., x_n\}$

$$Q_2 = \{x \in X | P(X > x) = P(X < x) = 0.5\}$$

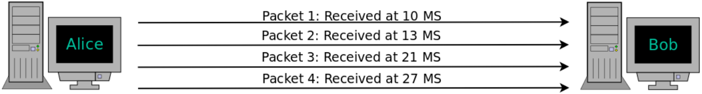
$$R_\psi(x_i) = r_i = \begin{cases} 1 & x_i > Q_2 \\ 0 & x_i < Q_2 \end{cases}$$

Hence, the entropy is extracted into the binary value: $r_1 r_2 r_3 r_4 ... r_n$

Note: alternative measures of center can be used in the place of Q_2 but only Q_2 maximizes the extracted entropy

A Posteriori Extractor for Inter-Packet Delays Example

Figure 6: Example Entropy Extraction (A Posteriori method)



$$T_1 = P_2 - P_1 = 13 - 10 = 3$$

$$T_2 = P_3 - P_2 = 21 - 13 = 8$$

$$T_3 = P_4 - P_3 = 27 - 21 = 6$$

$$Q_2 = 6$$

```
for T_i:
  if T_i > Q_2:
      record one
  else if T_i < Q_2:
      record zero
  else:
      record nothing
```

In this small example the extracted random string is $01 = 1$

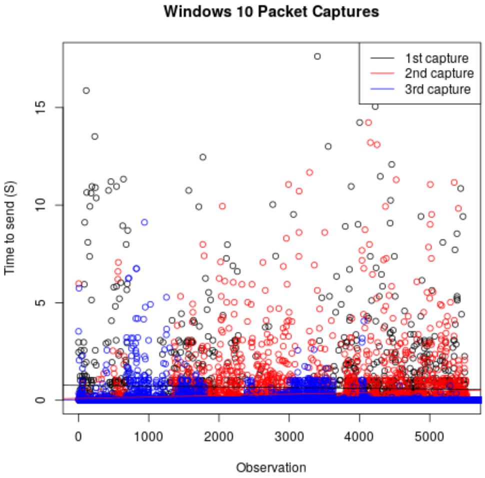
Experimental Set-Up

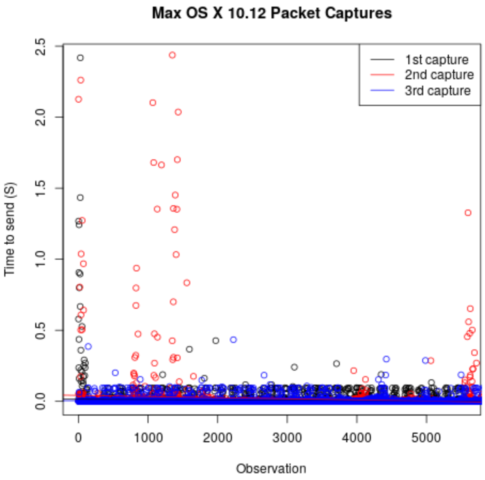
- Inter-Packet Timings: time differences between packet arrivals
- Arrival times (in μs) captured by Wireshark & TCPdump
- Five machines used:

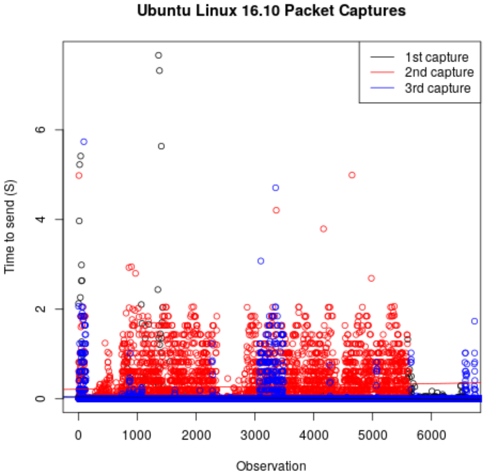
Machine	OS	CPUs	RAM	Speed
1	Windows 10	2	8 Gb	2.35 GHz
2	MacOS 10.12	2	8 Gb	2.6 GHz
3	Ubuntu 16.10	8	16 Gb	2.6 GHz
4	Ubuntu 17.04	8	16 Gb	2.8 GHz
5	Ubuntu 17.04	8	32 Gb	3.2 GHz

Outline

Initial Packet Capture Timings




Max OS X 10.12 Packet Captures
1st capture
2nd capture
3rd capture
Time to send (S)
0.0
0.5
1.0
1.5
2.0
2.5
0
1000
2000
3000
4000
5000
Observation


Ubuntu Linux 16.10 Packet Captures
1st capture
2nd capture
3rd capture
Time to send (S)
0
2
4
6
0
1000
2000
3000
4000
5000
6000
Observation

Before and After on an Idle Network

Figure 9: Idle Before Hashing

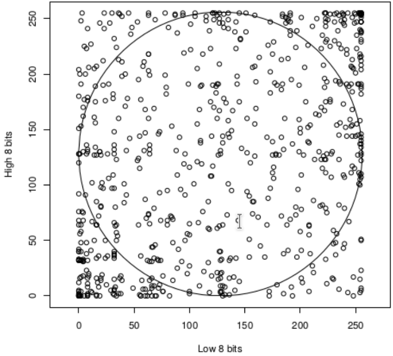
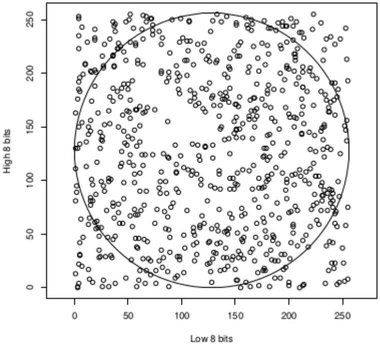


Figure 10: Idle After Hashing



Before and After on Busy Network

FIGURE 11: BUSY BEFORE HASHING

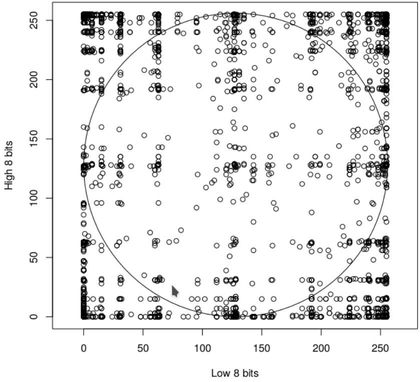
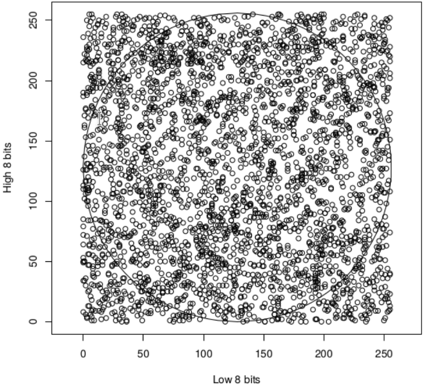
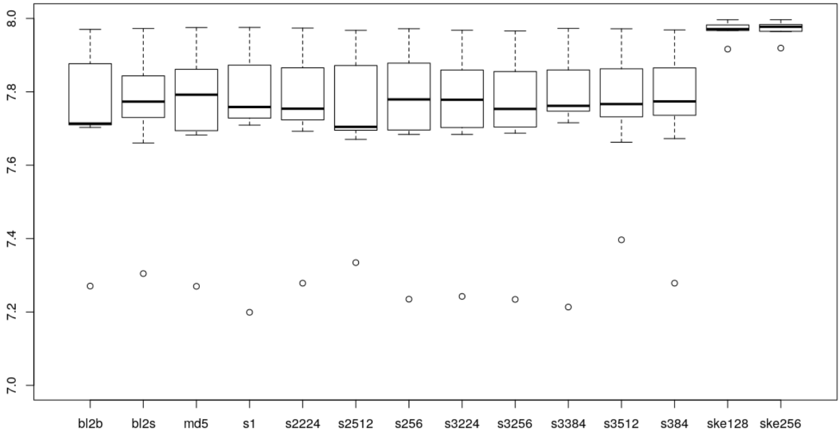


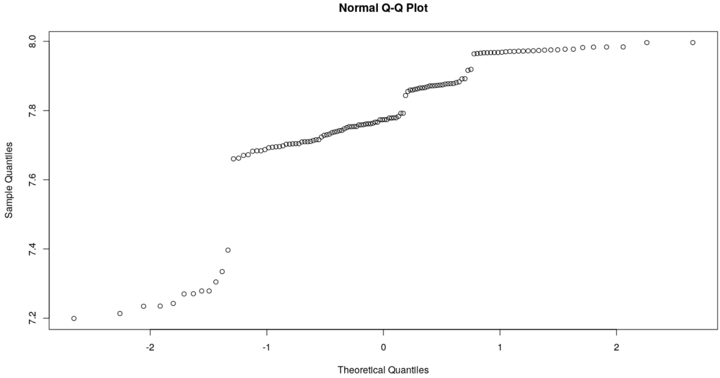
FIGURE 12: BUSY AFTER HASHING



Boxplot of Entropy Values for Common Hashes



Checking the ANOVA Assumptions for Entropy (Normality)



Shapiro Wilks Test for Normality (Reject Null that data are normal)

W	0.81796
p-val	3.418e-11**

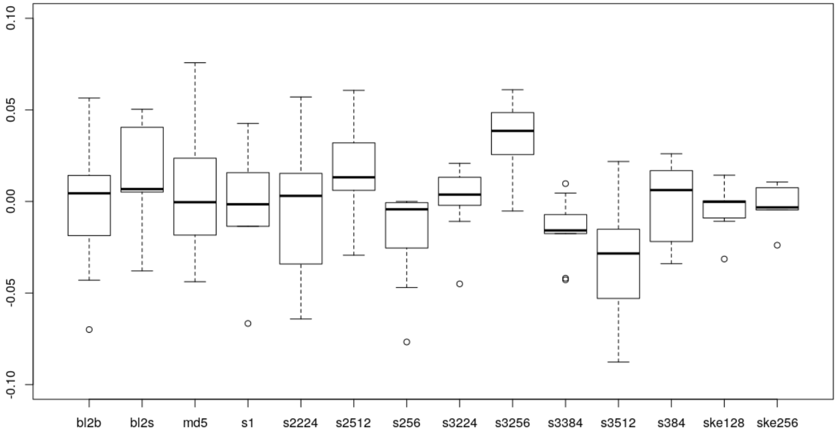
Kruskal-Wallis (Non Parametric ANOVA) Results

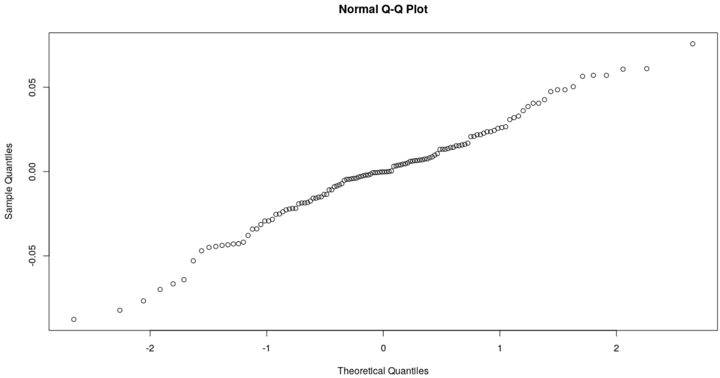
```
  Kruskal-Wallis rank sum test

data: x and group
Kruskal-Wallis chi-squared = 30.198, df = 13, p-value = 0

                        Comparison of x by group
                            (No adjustment)
Col Mean-|
Row Mean |       bl2b       bl2s        md5         s1
---------+--------------------------------------------
    bl2s |  -1.207029
         |     0.1137
         |
     md5 |  -0.464738   0.742291
         |     0.3211     0.2290
         |
      s1 |  -0.232369   0.974660   0.232369
         |     0.4081     0.1649     0.4081
```

Boxplot of Serial Correlations for Common Hashes





Shapiro Wilks Test for Normality (Accept Null that data are normal)

W	0.98486
p-val	0.1741

Checking the ANOVA Assumptions for SC (Homosce.)

Levene test for Homoscedasticity (Accept Null that data are HS)

F	1.4785
p-val	.1364

ANOVA Results for Serial Correlation

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = dfB$Serial.Correlation ~ dfB$Hash)

              Df  Sum Sq   Mean Sq F value   Pr(>F)
dfB$Hash      13 0.03032 0.0023321   2.938 0.00104 **
Residuals    112 0.08891 0.0007938


$`dfB$Hash`
                    diff          lwr          upr     p adj
bl2s-bl2b   0.0187737778 -0.026766335 0.0643138905 0.9784332
md5-bl2b    0.0115043333 -0.034035779 0.0570444460 0.9998304
s1-bl2b     0.0042363333 -0.041303779 0.0497764460 1.0000000
```

Outline

Primary Hypothesis

Hypothesis

Assuming a **cryptographic hash** is being used to increase the **apparent randomness** of a data set, It is possible to **formulate metrics** to choose the best hash for this purpose.

Conclusion

The hypothesis holds, and suitable metrics were formulated and verified.

Secondary Hypothesis

Secondary Hypothesis

The **A Posteriori** method described in this research is a valid approach for entropy extraction of a **weak random source** in the form of inter packet delays between packet arrivals.

Conclusion

The method proposed can indeed function as a randomness extractor on network timing data.

Future Steps

1. Perform analysis looking at different metrics (STS/DieHarder results)
2. Perform analysis with wider variety of initial strings from different sources.
3. Examine mean differences in theoretical light.
4. Apply analysis to more types of one-way functions.

Thankyou For your Time

QUESTIONS??

BACKUP SLIDES

A Posteriori Maximizes Shannon's Entropy (1)

[PROOF:]

Given a *supposedly* random sample

$$X = \{x_1 \in \mathbb{R}, x_2 \in \mathbb{R}, x_3 \in \mathbb{R}, \ldots, x_n \in \mathbb{R}\}$$

We define the random variable α in terms of the median (or second quartile) of X

$$\alpha : \mathbb{R} \to \mathbb{B}$$

$$P(\alpha = 0) = p_0(\alpha) = \frac{|\{x|x < Q_2(X)\}|}{|X|} = \frac{1}{2}$$

$$P(\alpha = 1) = p_1(\alpha) = \frac{|\{x|x > Q_2(X)\}|}{|X|} = \frac{1}{2}$$

The formula for the entropy of a string of Bernoulli trials (or a 'bitstring') is given:

$$H(p_0(b), p_1(b)) = -(p_0(b)\log_2(p_0(b)) + p_1(b)\log_2(p_1(b)))$$

We can maximize the Entropy function as so:

$$\nabla H(p_0, p_1) = \Big(\frac{\partial H}{\partial p_0}, \frac{\partial H}{\partial p_1}\Big) = \Big(-\frac{\ln(p_0) + 1}{\ln(2)}, -\frac{\ln(p_1) + 1}{\ln(2)}\Big)$$

Maximizing we find

$$\frac{-\ln(p_0) - 1}{\ln(2)} = 0 \implies \ln(p_0) = -1 \implies p_0 = \frac{1}{e}$$

$$\frac{-\ln(p_1) - 1}{\ln(2)} = 0 \implies \ln(p_1) = -1 \implies p_1 = \frac{1}{e}$$

A Posteriori Maximizes Shannon's Entropy (2)

This seemingly odd result is because there is an *inherent* dependence among these two values, expressed mathematically as $p_0 + p_1 = 1$, in our first maximization attempt, we neglected to account for the hard-restraint $p_0 + p_1 = 1$ In constraining the original optimization we have the following system:

$$\frac{-\ln(p_1) - 1}{\ln(2)} = 0 = \frac{-\ln(p_0) - 1}{\ln(2)}$$

$$p_1 = 1 - p_0$$

$$\frac{-\ln(1 - p_0) - 1}{\ln(2)} = \frac{-\ln(p_0) - 1}{\ln(2)} \implies 1 - p_0 = p_0$$

$$\implies p_0 = 0.5 \implies p_1 = 1 - 0.5 = 0.5$$

Because $p_0(\alpha) = p_1(\alpha) = 0.5$ by definition, we have maximized the entropy function for the constraint

$p_1 + p_0 = 1.$ $\square$

Entropy (1/4)

- **Entropy** is related to the idea of **self-information**, but the two **are not** synonymous.
- The self-information of a particular event is a measure of how much information is contained by that event occurring.
- Events that occur more frequently have lower self-information.
- Self-Information is inversely proportional the the frequency of an event.
- Intuitively, we may define it as the following:

$$A \in S \implies I(A) = \frac{1}{P(A)} = \frac{1}{\frac{\|A\|}{\|S\|}} \tag{1}$$

Entropy (2/4)

- This measure is not additive under the intersection operator.
- in other words, the self information of event B + the self information of event A should be equivalent to the self information of the intersection of A and B.
- We can see that our intuitive definition does not satisfy this property.

$$(I(A) = \frac{1}{P(A)}) \wedge (I(B) = \frac{1}{P(B)}) \tag{2}$$

$$\implies I(A \cap B) = \frac{1}{P(A) \cdot P(B)} \tag{3}$$

$$I(A) + I(B) = \frac{P(A) + P(B)}{P(A) \cdot P(B)} \neq \frac{1}{P(A) \cdot P(B)} \tag{4}$$

Entropy (3/4)

- Hence our intuitive definition of self information does not satisfy the additive property.
- We are moved to consider a different measure

$$I(A) = ln\left(\frac{1}{P(A)}\right) I(B) = ln\left(\frac{1}{P(B)}\right) \tag{5}$$

$$I(A \cap B) = ln\left(\frac{1}{P(A) \cdot P(B)}\right) \tag{6}$$

$$I(A) + I(B) = ln\left(\frac{1}{P(A)}\right) + ln\left(\frac{1}{P(B)}\right) = ln\left(\frac{1}{P(A) \cdot P(B)}\right) \tag{7}$$

Entropy (4/4)

- So we now have a definition of self-information.
- Because this definition is based on the pmf it is a random variable
- As a random variable we can take the expected value

$$H(X) = E(I(X)) \tag{8}$$

- The above measure is known as the entropy of an event X.
- So we can calculate the entropy of a bit string as:

$$H(X) = -\sum_{i=0}^{n} P(X)I(X) \tag{9}$$

$$= -(P(X=0)lg(P(X=0)) + P(X=1)lg(P(X=1))) \tag{10}$$

Entropy Example

- As an example of bitstring entropy calculation consider the bitstring 11011010110110111011110001010101101
- There are 35 bits in the bit string, 22 of which are 1's and 13 of which are 0's
- $P(X=1) = 0.628571429$
- $P(X=0) = 0.371428571$
- $H(X) = -(-0.9517626753) = 0.9517626753$